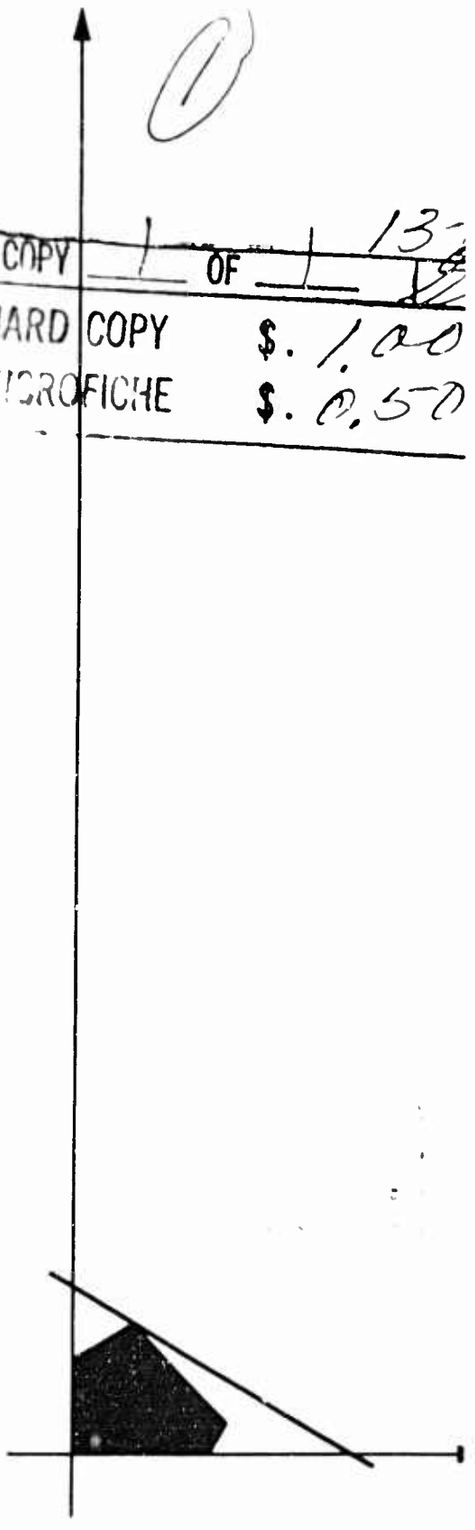
ORC 64-33
DECEMBER 1964

UPDATING THE PRODUCT FORM OF THE INVERSE FOR THE REVERSED SIMPLEX METHOD

by

George B. Dantzig
R. P. Harvey
R. D. McKnight



OPERATIONS RESEARCH CENTER

INSTITUTE OF ENGINEERING RESEARCH

UNIVERSITY OF CALIFORNIA - BERKELEY

UPDATING THE PRODUCT FORM OF THE INVERSE

FOR THE REVISED SIMPLEX METHOD

by

G. B. Dantzig
University of California at Berkeley
R. P. Harvey and R. D. McKnight
Standard Oil Company of California

December 1964

ORC 64-33

The research of Roy Harvey and Robert McKnight was directly supported by Standard Oil Company of California. The research of George B. Dantzig was partly supported by O.N.R. Contract Nonr-222(83).

UPDATING THE PRODUCT FORM OF THE INVERSE FOR THE REVISED SIMPLEX METHOD

A method by which the number of transformation matrices in the product form of the inverse for the revised simplex method of solution of linear programs need never exceed m, the number of rows.

Computer codes for solving linear programs by the simplex method usually use one of three forms in representing the problem during the course of solution. These are (a) the standard form or original simplex method, (b) the revised simplex method with explicit inverse; and (c) the revised simplex method with inverse in product form (ref. 1). [For a comparison of the relative efficiences of the three methods, see ref. 2.] It is hoped that the method to be proposed will at least partially alleviate one of the principal disadvantages of (c), the product form algorithm, namely the need for frequent re-inversion of the basis in order to reduce the number of transformations, without sacrificing too much of some of its advantages, such as the sparseness of the inverse and the ease with which the inverse is kept current. The chief advantage of this proposal is that the number of non-zeros in the inverse representation is conserved and remains approximately constant after the initial build up.

The product form of an inverse with which we are concerned here is the indicated product of a number of elementary $m \times m$ transformation matrices, each such matrix being an identity matrix with the exception of one column, the so-called transformation column. Computer codes

using the product form of the inverse need to store only these exceptional columns and their column indices (usually only the non-zero elements of these columns and their row positions are stored).

The normal procedure is to adjoin an additional elementary matrix to this product every simplex iteration until such time as their number becomes too large for efficiency or the limit of storage capacity is reached; at which time the current basis matrix is re-inverted and k , $k \leq m$, new transformation matrices are formed. For our purposes, the inverse of the basis B is represented by the product of k elementary transformation matrices T_t :

$$B^{-1} = T_k \, T_{k-1} \cdots T_1$$

Let us assume that these k transformation matrices $T_1, T_2, \ldots, T_k$ correspond to columns $Q_1, Q_2, \ldots, Q_k$ which have replaced, in turn, each of the unit vectors $U_{h_1}, U_{h_2}, \ldots, U_{h_k}$ in the initial unit basis. For the purpose of exposition, only, suppose that the rows have been reordered so that $h_t = t$ (of course, this is not necessary in practice); call $U_{h_t} = V_t$. The representation of an outgoing column V_t in terms of a basis $Q_1, Q_2, \ldots, Q_{t-1}, Q_t, V_{t+1}, \ldots, V_m$ yields the usual system of product form relations:

$$(1) \qquad V_t = \tau_{1,t} Q_1 + \tau_{2,t} Q_2 + \ldots + \tau_{t,t} Q_t + \tau_{t+1,t} V_{t+1} + \ldots + \tau_{m,t} V_m$$

where $\tau_{t,t} \neq 0$, $t=1,2,\ldots,k$.

If, on the other hand, we denote by $\alpha_{i,t}$ the coefficients resulting when relation (1) is solved for Q_t , that is:

$$\alpha_{i,t} = -\tau_{i,t}/\tau_{t,t} \qquad , \ i \neq t$$

$$\alpha_{t,t} = 1/\tau_{t,t}$$

we have an alternative way to express the system of product form relations:

$$(2) \qquad Q_t = \alpha_{1,t}Q_1 + \alpha_{2,t}Q_2 + \ldots + \alpha_{t,t}V_t + \alpha_{t+1,t}V_{t+1} + \ldots + \alpha_{m,t}V_m$$

where $\alpha_{t,t} \neq 0$, $t=1,2,\ldots,k$.

Relation (2), which is the one we use in this paper, is the representation of an in-going column Q_t in terms of the basis $Q_1, Q_2, \ldots, Q_{t-1}, V_t, V_{t+1}, \ldots, V_m$.

Now, suppose that a new vector Q_{k+1} is chosen to enter the basis. If it were to replace one of the unit vectors still in the basis, we will adjoin the transformation to product form (2) in the usual way. We will not concern ourselves further with this case. If, however, it were to replace Q_r , then $\alpha_{r,k+1} \neq 0$, and we have the following:

$$\begin{aligned}
Q_1 &= \alpha_{1,1}V_1 + \alpha_{2,1}V_2 + \ldots + \alpha_{r,1}V_r + \ldots + \alpha_{k,1}V_k + \ldots + \alpha_{m,1}V_m \\
Q_2 &= \alpha_{1,2}Q_1 + \alpha_{2,2}V_2 + \ldots + \alpha_{r,2}V_r + \ldots + \alpha_{k,2}V_k + \ldots + \alpha_{m,2}V_m \\
&\vdots \\
Q_{r-1} &= \alpha_{1,r-1}Q_1 + \alpha_{2,r-1}Q_2 + \ldots + \alpha_{r,r-1}V_r + \ldots + \alpha_{k,r-1}V_k + \ldots + \alpha_{m,r-1}V_m \\
(3) \quad Q_r &= \alpha_{1,r}Q_1 + \alpha_{2,r}Q_2 + \ldots + \alpha_{r,r}V_r + \ldots + \alpha_{k,r}V_k + \ldots + \alpha_{m,r}V_m \\
Q_{r+1} &= \alpha_{1,r+1}Q_1 + \alpha_{2,r+1}Q_2 + \ldots + \alpha_{r,r+1}Q_r + \ldots + Q_{k,r+1}V_k + \ldots + \alpha_{m,r+1}V_m \\
&\vdots \\
Q_k &= \alpha_{1,k}Q_1 + \alpha_{2,k}Q_2 + \ldots + \alpha_{r,k}Q_r + \ldots + \alpha_{r,k}Q_r + \ldots + \alpha_{m,k}V_m \\
Q_{k+1} &= \alpha_{1,k+1}Q_1 + \alpha_{2,k+1}Q_2 + \ldots + \alpha_{r,k+1}Q_r + \ldots + \alpha_{k,k+1}Q_k + \ldots + \alpha_{m,k+1}V_m
\end{aligned}$$

where $\alpha_{t,t} \neq 0$ for $t=1,2,\ldots,k$ by (2).

Our objective is to eliminate Q_r using only a small amount of additional working storage. Relation r is used as a starting "eliminator" and relations r+1, r+2,...,k+1 are up-dated, resulting in a new product form with just k transformations, as shown in (4):

$$
\begin{aligned}
Q_1 &= \alpha_{1,1} V_1 + \alpha_{2,1} V_2 + \ldots + \alpha_{r,1} V_r + \alpha_{r+1,1} V_{r+1} + \ldots + \alpha_{k,1} V_k + \ldots + \alpha_{m,1} V_m \\
Q_2 &= \alpha_{1,2} Q_1 + \alpha_{2,2} V_2 + \ldots + \alpha_{r,2} V_r + \alpha_{r+1,2} V_{r+1} + \ldots + \alpha_{k,2} V_k + \ldots + \alpha_{m,2} V_m \\
&\vdots \\
Q_{r-1} &= \alpha_{1,r-1} Q_1 + \alpha_{2,r-1} Q_2 + \ldots + \alpha_{r,r-1} V_r + \alpha_{r+1,r-1} V_{r+1} + \ldots + \alpha_{k,r-1} V_k + \ldots + \alpha_{m,r-1} V_m \\
Q_{r+1} &= \alpha'_{1,r} Q_1 + \alpha'_{2,r} Q_2 + \ldots + \alpha'_{r,r} C_r + \alpha'_{r+1,r} F_{r+1} + \ldots + \alpha'_{k,r} V_k + \ldots + \alpha'_{m,r} V_m \\
Q_{r+1} &= \alpha'_{1,r+1} Q_1 + \alpha'_{2,r+1} Q_2 + \ldots + \alpha'_{r,r+1} Q_{r+1} + \alpha'_{r+1,r+1} C_{r+1} + \ldots + \alpha'_{k,r+1} V_k + \ldots + \alpha'_{m,r+1} V_m \\
&\vdots \\
Q_{k+1} &= \alpha'_{1,k} Q_1 + \alpha'_{2,k} Q_2 + \ldots + \alpha'_{r,k} Q_{r+1} + \alpha'_{r+1,k} Q_{r+2} + \ldots + \alpha'_{k,k} C_k + \ldots + \alpha'_{m,k} V_m
\end{aligned}
\tag{4}
$$

where $\alpha'_{t,t} \neq 0$ for $t=r,\ldots,k$ and for $t \geq r$ either (C_t, F_{t+1}) is the same as (F_t, V_{t+1}) or (V_{t+1}, F_t), where F_r is the same as V_r. Before giving a description of the algorithm, which is oriented towards a computer code adaptation, it might be informative to consider an example. It will be helpful here to assume that all of the relations involved are expressed in the form of (2). For the sake of some generality, however, we will not reorder the rows. Therefore, the unit vectors will be the U_{h_t} and the pivot positions will not necessarily be on the diagonal. We will indicate the pivots by underlining. The example shows the original array, a "motion picture" of the step-by-step procedure, and finally the updated array. Each step consists of updating a relation by the application of an eliminator, and then forming a revised eliminator to be used on the next step by the use of the updated relation. The

initial eliminator is just the original expression for Q_1. An updated relation is allowed to pivot either on the same index position as that underlined in the original relation or the position underlined in the current eliminator (the next eliminator will always have its underline in the alternative position to the one chosen for pivot). We will choose the one with the largest coefficient in absolute value, making an arbitrary choice in case of ties.

ORIGINAL RELATIONS

$$Q_1 = U_1 + \underline{2U_2} - U_3 + 0U_4$$

$$Q_2 = -U_1 - \underline{Q_1} - U_3 + \underline{U_4}$$

$$Q_3 = \underline{U_1} + 0Q_1 + U_3 - Q_2$$

$$Q_4 = Q_3 - \underline{Q_1} + \underline{U_3} - Q_2$$

$$Q_5 = 0Q_3 + \underline{Q_1} + 0Q_4 + Q_2$$

ENTER DROP

1st ELIMINATOR: $Q_1 = U_1 + \underline{2U_2} - U_3$

OLD: $Q_2 = -U_1 - \underline{Q_1} - U_3 + \underline{U_4}$

Step 1. ELIMINATE Q_1:

UPDATED $Q_2 = -2U_1 - \underline{2U_2} + U_4$

ELIMINATE U_2 from the eliminator

FORM 2nd ELIM:	$Q_1 = -U_1 - Q_2 - U_3 + \underline{U_4}$
OLD:	$Q_3 = \underline{U_1} + U_3 - Q_2$

Step 2. ELIMINATE Q_1 from the eliminator

UPDATED $Q_3 = \underline{U_1} - Q_2 + U_3$

ELIMINATE U_1:

FORM 3rd ELIM:	$Q_1 = -Q_3 - 2Q_2 + \underline{U_4}$
OLD:	$Q_4 = Q_3 - Q_1 + \underline{U_3} - Q_2$

Step 3. ELIMINATE Q_1 from the eliminator

UPDATED $Q_4 = 2Q_3 + Q_2 + \underline{U_3} - U_4$

ELIMINATE U_3:

FORM 4th ELIM:	$Q_1 = -Q_3 - 2Q_2 + \underline{U_4}$
OLD:	$Q_5 = \underline{Q_1} + Q_2$

Step 4. ELIMINATE Q_1 from the eliminator

UPDATED $Q_5 = -Q_3 - Q_2 + \underline{U_4}$

UPDATED RELATIONS

$$Q_2 = -2U_1 - \underline{2U_2} + U_4$$

$$Q_3 = \underline{U_1} - Q_2 + U_3$$

$$Q_4 = 2Q_3 + Q_2 + \underline{U_3} - U_4$$

$$Q_5 = -Q_3 - Q_2 + \underline{U_4}$$

This algorithm is one of several possible variants for updating the transformations. This one happens to proceed through the transformations in a forward direction. Backward working algorithms can be formulated along very similar lines.

Method

Assume that in addition to the foregoing k transformations α^t t=1,2,...,k , that the following properly ordered aways are given:

J ; J_i = column name associated with basic variable with unit coefficient on row i of the canonical form.

X ; x_{J_i} = current value of basic variable.

F ; f_i = storage containing the entering column (called j^*) expressed in terms of the current basis.

H ; h_t = pivot row corresponding to the t^{th} transformation.

The following arrays are available for working storage:

D ; δ_i = for updating α^t .

E ; e_i = for updating eliminators.

L ; ℓ_i = for recording new pivot positions during updating.

The following one-cell arrays are given:

j^* = name of column entering the basic set.

k = number of transformations α^t.

r = pivot row corresponding to the $(k+1)^{st}$ transformation F.

and the following one-cell arrays are available for working storage:

t = running index on transformations.

p = pivot position of the updated α^t.

q = alternate pivot position not selected above.

We will use superscripts with arrays D and E to differentiate their contents during the updating steps.

Initialization

(A) Transform X : $x'_{j_r} = \frac{x_r}{f_r}$; $x'_{j_i} = x_{j_i} - x'_{j_r} f_i$, $i \neq r$

(B) Replace J_r by new column name j^*.

(C) L : $L_i = i$

(D) Find $t = s$ such that $h_{s-1} = r$; if none store α^{k+1} where $\alpha_{i,k+1} = f_i$ with $h_{k+1} = r$ and go to (i)

(E) Set E^t : $e_i^t = \alpha_{i,t-1}$

(F) Set $q = r$

Algorithm

(a) If $t = k+1$ go to (g) otherwise Form $D^{(t)}$:

$$\delta_q^t = \alpha_{r,t} e_q^t \quad \text{and} \quad \delta_i^t = \alpha_{L_i,t} + \alpha_{r,t} e_i^t \quad \text{for} \quad i \neq q$$

(b) Choose new $p = q$; new $q = h_t$ if $|\delta_q^t| > |\delta_{h_t}^t|$ otherwise new $p = h_t$ and q remains unchanged

(c) Store updated α^{t-1} where $\alpha_{i,t-1} = \delta_i^t$ with updated $h_{t-1} = p$

(d) If $h_t = p$ go to (e); otherwise exchange x'_{h_t} with x'_p

J_{h_t} with J_p

(e) Form E^{t+1} : $e_p^{t+1} = e_p^t / \delta_p^t$; $e_i^{t+1} = e_i^t - e_p^{t+1} \delta_i^t$, $i \neq p$

(f) Update $L_p = h_t$; increase t by 1 , go to (a)

(g) Form $D^{(t)}$: $\delta_q^t = f_r e_q^t$; $\delta_i^t = f_{L_i} + f_r e_i^t$ for $i \neq q$

(h) Store updated α^k where $\alpha_{i,k} = \delta_i^t$ with updated $h_k = q$

(i) Exit.

Proof that there is always at least one non-zero element

This algorithm on each updating step, except that last, makes a choice from one of two positions for the pivot location. It is established below that at least one of these two positions has a non-zero coefficient.

Following and adding to the notation of the algorithm described earlier, denote by $p(t)$ and $q(t)$ the p^{th} and q^{th} position choices at the t^{th} stage.

Suppose that the value of t for which $h_{t-1} = r$ is s so that $t = s , s+1,\ldots,k+1$ in the steps of the algorithm. Consider any value of t such that $s \leq t \leq k$. Let us make the inductive assumption that $e^t_{q(t-1)} \neq 0$.

From a) $\delta^t_{h_t} = \alpha_{h_t,t} + \alpha_{r,t} e^t_{h_t}$, $\alpha_{h_t,t} \neq 0$

$$\delta^t_{q(t-1)} = \alpha_{r,t} e^t_{q(t-1)}$$

If $\alpha_{r,t} = 0$ then $\delta^t_{r_t} \neq 0$ since the old pivot $\alpha_{h_t,t} \neq 0$; if $\alpha_{r,t} \neq 0$, then $\delta^t_{q(t-1)} \neq 0$ by our inductive assumption. Hence there exists at least one non-zero pivot choice. We need also to show $e^{t+1}_{q(t)} \neq 0$. There are two cases to consider:

Case 1: Suppose $q(t) = h_t$ and $p(t) = q(t-1)$ implying $\alpha_{r,t} \neq 0$. The elimination of $q(t-1)$ term from the eliminator yields (see Step e)

$$e^{t+1}_{q(t)} = e^t_{h_t} - \frac{e^t_{q(t-1)}(\alpha_{h_t,t} + \alpha_{r,t} e^t_{h_t})}{\alpha_{r,t} e^t_{q(t-1)}} = \frac{-\alpha_{h_t,t}}{\alpha_{r,t}} \neq 0$$

Case 2: Otherwise $p(t) = h_t$ and $q(t) = q(t-1)$ implying similarly $\alpha_{h_t,t} + \alpha_{r,t} e^t_{h_t} \neq 0$. In this case from (e) again

$$e^{t+1}_{q(t)} = e^t_{q(t-1)} - \frac{e^t_{h_t} \alpha_{r,t} e^t_{q(t-1)}}{\alpha_{h_t,t} + \alpha_{r,t} e^t_{h_t}} = \frac{e^t_{q(t-1)} \alpha_{h_t,t}}{\alpha_{h_t,t} + \alpha_{r,t} e^t_{h_t}} \neq 0$$

To complete the inductive step we note that initially for $t=s$, $e^s_{q(s-1)} = e^s_r \neq 0$, hence by induction $e^{t+1}_{q(t)} \neq 0$ and also the coefficient of the pivot $\delta^t_{p(t)} \neq 0$ for $t=s,s+1,\ldots, k$.

Finally at step $k+1$ the pivot index is $q(k)$ by (g) . By definition $f_r = \alpha_{r,k+1}$ and $f_r \neq 0$. Thus, from our inductive hypothesis

$$\delta_{q(k)}^{k+1} = \alpha_{r,k+1} e_{q(k)}^{k+1} \neq 0$$

so that the final pivot coefficient is also non-zero completing our proof.

The procedure could be generalized to give more freedom as regards choice of pivot at the expense of carrying along more eliminators. As noted earlier, it is possible to have backward elimination as well as forward.